Planning as Heuristic Search for Incremental Fault Diagnosis and Repair

Håkan Warnquist* and **Jonas Kvarnström** and **Patrick Doherty**
Dept. of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden
{g-hakwa, jonkv, patdo}@ida.liu.se

Abstract

In this paper we study the problem of incremental fault diagnosis and repair of mechatronic systems where the task is to choose actions such that the expected cost of repair is minimal. This is done by interleaving acting with the generation of partial conditional plans used to decide the next action. A diagnostic model based on Bayesian Networks is used to update the current belief state after each action. The planner uses a simplified version of this model to update predicted belief states. We have tested the approach in the domain of troubleshooting heavy vehicles. Experiments show that a simplified model for planning improves performance when troubleshooting with limited time.

1 Introduction

Modern mechatronic systems are complex products integrating electronics, mechanics and software. Due to their intricate architecture and functionality they are often difficult for a technician to troubleshoot. Efficient computer aided troubleshooting that optimizes the expected cost of repair may yield large cost savings for the industry.

A good example is the troubleshooting of vehicles. When a vehicle enters the workshop the mechanic has only partial information of the vehicle's true state, but given the information available a probability distribution over possible faults can be inferred. The mechanic may choose to perform actions to acquire more information or to repair suspected components. With a model describing the probabilistic dependencies between observations and faults as well as the costs, effects and preconditions of the actions, the troubleshooting problem can be viewed as a probabilistic planning problem with incomplete information.

For non-trivial problem instances, we generally cannot afford to compute complete conditional troubleshooting plans with minimal expected costs. However, the mechanic only requires information about the next action to perform. We take advantage of this fact and use a heuristic anytime algorithm that incrementally decreases expected cost of the plan, but can be interrupted at any time, yielding an initial action to perform. By interleaving planning and acting, and taking into consideration new information gained by each action, the search space can be incrementally constrained and high quality decisions can be made at every step.

We propose to use different models of the domain when we plan and when we act. A precise model of the domain is used to update the belief of the current world state given observations and actions that the mechanic has performed. When we plan, a filtered version of the precise model is used where strong probabilistic dependencies close to 0 or 1 are made deterministic. With the filtered model, planning is still probabilistic and contingent, but the expansion of new states in a search graph is faster. This allows the planner to provide more support for each decision when the domain is large and there is little time for computation.

As a motivating example we have studied the domain of troubleshooting heavy trucks. This is a highly relevant industrial domain where any improvement in troubleshooting performance may lead to significant savings in terms of time and money, both for workshops and for haulage contractors. In this domain, we also show how the search space can be reduced by on-demand generation of composite actions, allowing the planner to focus on observation and repair actions that affect our beliefs about faulty components as opposed to actions that prepare the vehicle for observations and repairs. To evaluate the effects of filtering the model, we have implemented an incremental troubleshooter using conventional planning techniques. Preliminary empirical experiments show that our filtered model improves the quality of decisions made and that it is robust against small errors in the parameters.

The outline of this paper is as follows. We begin by describing the heavy vehicles domain in Section 2 Then in Section 3 we will go through the architecture of our troubleshooting system. In Sections 4 and 5 we describe the modeling of the domain and how the model used for planning is created. In Section 6 we describe the planner and the heuristics used for the empirical evaluation in Section 7 We end by mentioning some related work in Section 8 before the conclusion in Section 9

2 Troubleshooting Heavy Vehicles

Heavy vehicles are representative mechatronic systems composed of many components and interacting subsystems. Modern vehicles are often required by law to be equipped with an on-board diagnostics system (OBD) (U.S Environmental Protection Agency 2009). The task of the OBD is to generate alarms when faults are suspected. These alarms are often created when an observed sensor value differs from the expected where the thresholds are designed so that the risk of false alarms is low. The OBD is not always powerful enough to specify exactly the faulty components, but if any component is faulty at least one alarm will trigger.

In the workshop, the mechanic has access to further observations of varying specificity. Some observations are symptoms that may be caused by several components while others

*Affiliated with Scania CV AB.

are direct inspections of component statuses. The dependency between a component and an observation is not deterministic. For example, when a control valve breaks, it may or may not cause bad braking performance. On the other hand, observations have low variance: If we have already observed bad braking performance, and we check the braking performance again without repairing any of its possible causes (such as the control valve), we expect to observe the same result.

To reach some parts of the vehicle, one must first disassemble other parts. The level of disassembly is the *configuration state* of the vehicle. Each action can only be performed in certain configuration states corresponding to constraints on the parts of the vehicle that must be assembled or disassembled. Actions the mechanic may choose from are observations, repairs and changes to the configuration. Repairing a component means that the component is replaced. It is possible but unlikely that a repair fails. When the mechanic is finished troubleshooting, the vehicle must be taken on a test drive to verify that it functions properly and that the OBD triggers no more alarms. During the test drive the mechanic may make other observations such as listening for noise or feeling how the vehicle behaves.

3 Architecture of the Troubleshooting System

The troubleshooting system shown in Figure 1 consists of five parts: Two diagnostic engines (D1 and D2), an observer (O), an action proposer (AP) and a planner (P). Using different models, the task of the diagnostic engines is to infer a new belief state, i.e. a probability distribution over component statuses, given the outcome of an action. The action proposer and the observer are the interfaces to the user. The action proposer recommends actions to the user and the observer maintains the current belief state and configuration of the system we troubleshoot given the outcomes of the actions that have been performed by the user. When updating the belief state, D1 uses the precise model of the system while D2 uses a simplified model instead. The planner uses this diagnostic engine to predict resulting belief states and their likelihoods given the possible outcomes of future actions.

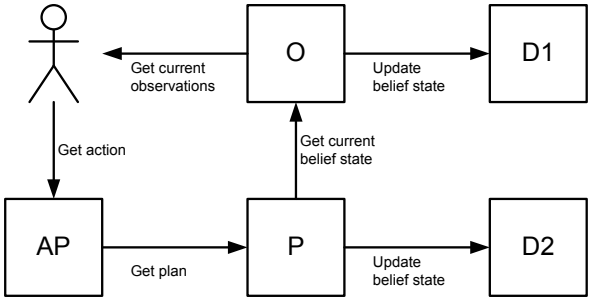


Figure 1: The troubleshooting system.

Troubleshooting starts by the user querying the action proposer to recommend an action. The action proposer then instantiates the planner to start planning from the current belief state given by the observer. When the user times out or the planner has found an optimal plan, the current best plan is returned to the action proposer which extracts the first action in the plan and recommends it to the user. After the user has performed the action, the outcome of that action is fed back to the observer which updates the next state using D1. The troubleshooting session terminates when the user verifies that the system functions properly by performing a successful *function control*. In the heavy vehicles domain a successful function control is a test drive of the vehicle where no alarms from the OBD are triggered. If the performed action is not a function control or if the function control fails, the action proposer initiates the planner again with the new state.

4 Modeling of the Domain

In this section we will describe the full model and how it is used to infer new belief states. The model has two parts: A diagnostic model and an action model.

4.1 Diagnostic Model

The diagnostic model describes the relation between the vehicle's components and the observations that can be made. We define components to be subsystems of the vehicle that can be replaced or repaired. Each component $i \in [1, N]$ is represented by a variable C_i that can take on the values non-faulty NF or faulty F. Each possible observation $j \in [1, M]$ is represented by an observation variable O_j that can take on the values non-indicating NI or indicating I.

We use a Bayesian Network (BN) to model the probabilistic dependencies between component and observation variables (see for example Jensen, 2001). A BN is a directed acyclic graph where variables are represented by nodes and dependencies are represented by directed edges. In the heavy vehicles domain, the BN has two layers of nodes, where the topmost layer consists of component variables and the bottom layer of observation variables. All edges are directed from component variables to observation variables. Each variable has a Conditional Probability Table (CPT) containing parameters describing the probability that a variable X takes on a specific value given the values of its parents $pa(X)$ in the BN, $P(X|pa(X))$. For example,

$$
\begin{aligned}
&P(\text{bad braking} = I | \text{ctrl valve} = NF, \text{pump} = NF) &&= 0.01\\
&P(\text{bad braking} = I | \text{ctrl valve} = NF, \text{pump} = F\ \ ) &&= 0.9\\
&P(\text{bad braking} = I | \text{ctrl valve} = F\ \ , \text{pump} = NF) &&= 0.6\\
&P(\text{bad braking} = I | \text{ctrl valve} = F\ \ , \text{pump} = F\ \ ) &&= 0.96
\end{aligned}
$$

The parameters in the CPT:s are assigned using expert knowledge, manufacturer's specifications and statistical data. To reduce the number of parameters needed to be set, all CPT:s are of the type *noisy-or* (Jensen 2001) unless special dynamics need to be expressed. The noisy-or CPT only needs the conditional probabilities for each single fault case and assumes positive interaction between multiple faults.

4.2 Making Inference in the Diagnostic Model

The diagnostic engine uses the diagnostic model to compute a *belief state* given an ordered set of evidence. Each piece of evidence is indexed by its temporal order and it can either be that a specific component is repaired or that a variable in the BN is observed to have a specific value. The belief state $\mathbf{b}_t$ at time t is a probability distribution over component statuses given all accumulated evidence $\mathbf{e}_{1:t}$ where each element $b_t(\mathbf{c}) \in \mathbf{b}_t$ is

$$b_t(\mathbf{c}) = P(\mathbf{c}_t | \mathbf{e}_{1:t}) \quad (1)$$

and $\mathbf{c}_t = (c_1, \ldots, c_N)$ is an assignment of all component statuses at time t.

Unless a repair is made at time t, it is assumed that the component statuses remain unchanged from time $t-1$, i.e.

$$P(\mathbf{c}_t | \mathbf{e}_{1:t-1}) = P(\mathbf{c}_{t-1} | \mathbf{e}_{1:t-1}) = P(\mathbf{c} | \mathbf{e}_{1:t-1}) \quad (2)$$

When the evidence at time t is an observation, $e_t = \{O_i^t{=}o\}$, the belief state is updated according to Bayes' rule.

$$b_t(\mathbf{c}) = \frac{P(O_i^t{=}o|\mathbf{c}, \mathbf{e}_{1:t-1}) b_{t-1}(\mathbf{c})}{\rho_{\mathbf{e}_{1:t}}} \quad (3)$$

where $\rho_{\mathbf{e}_{1:t}}$ is a normalization constant designating the likelihood of the observation e_t given $\mathbf{e}_{1:t-1}$.

A repair of a component C_i has to be treated differently from just observing that $C_i = NF$ since the repair *causes* the component to be non-faulty. Using the notation from (Pearl 2000) we denote the evidence that a repair is made at time t as $e_t = do(C_i^t{=}NF)$, and the belief state is updated by moving probability mass from a state where $C_i = F$ to one where $C_i = NF$,

$$b_t(\mathbf{c}) = \begin{cases} \varphi_i(b_{t-1}(\mathbf{c}) + b_{t-1}(\mathbf{c}')) & \text{if } c_i = F \\ (1-\varphi_i)(b_{t-1}(\mathbf{c}) + b_{t-1}(\mathbf{c}')) & \text{if } c_i = NF \end{cases} \quad (4)$$

where $\mathbf{c}' = (c'_1, \ldots, c'_N)$, $c'_j = c_j$ for $j \neq i$, and $c'_i = F$ and φ_i is the probability that the repair fails and the action instead causes the component to be faulty. Each probability φ_i $i \in [1, N]$ is known à priori and can be regarded as the probability that the replacement component is faulty from the factory.

For the heavy vehicle domain, we model the observations with no variance. If there is previous evidence for O_i and no repair has been made to any of the parents of O_i in the BN at a later time, this repeated observation will give the same result, i.e. if

$$\exists e_{t'} \nexists e_{t''} \in \mathbf{e}_{1:t-1}, 0 \leq t' < t'' \leq t-1,$$
$$e_{t'} = \{O_i^{t'}{=}o'\}, e_{t''} = do(C_j^{t''}{=}NF), C_j \in pa(O_i) \quad (5)$$

then

$$P(O_i^t{=}o|\mathbf{c}, \mathbf{e}_{1:t-1}) = \begin{cases} 1 & \text{if } o' = o \\ 0 & \text{if } o' = \neg o \end{cases} \quad (6)$$

otherwise the conditional probability of making the observation is looked up in its CPT:

$$P(O_i^t{=}o|\mathbf{c}^t, \mathbf{e}_{1:t-1}) = P(O_i{=}o|\mathbf{c}) \quad (7)$$

4.3 Configuration State

The degree of disassembly of the system is described by the configuration state which is completely observable. Each disassemblable part of the system is represented by a configuration variable that can be in one of the modes $assembled$ or $disassembled$. These variables relate to each other according to a directed acyclic graph describing how the disassemblable parts are attached to each other. For a configuration variable K to be $assembled$ all its children also needs to be $assembled$ and for K to be $disassembled$ all its parents must be $disassembled$. An example of this graph for a hydraulic braking system is shown in Figure 2. For example if we want to remove the propeller shaft K7 the oil in the retarder K1 and the oil in the gearbox K4 must be drained.

4.4 Action Model

Each action is modeled with a *cost*, a set of *preconditions*, and an ordered set of *effects*. The cost is a constant corresponding to the amount of time required to perform the action and the amount of resources consumed. The preconditions are conjunctions of expressions of the type $K = k$ where K is a configuration variable and $k \in \{assembled, disassembled\}$. To perform an action, all its preconditions must be true. The effects can be to repair a component, $do(C{=}NF)$, to observe the value of an observation variable, $observe(O)$, or to change the mode of a configuration variable, $do(K{=}k)$. The effects are ordered since it makes a difference if an observation is made before a repair or after. Each action that has effects that change the mode of configuration variables has preconditions that are consistent with the constraints defined by the configuration graph.

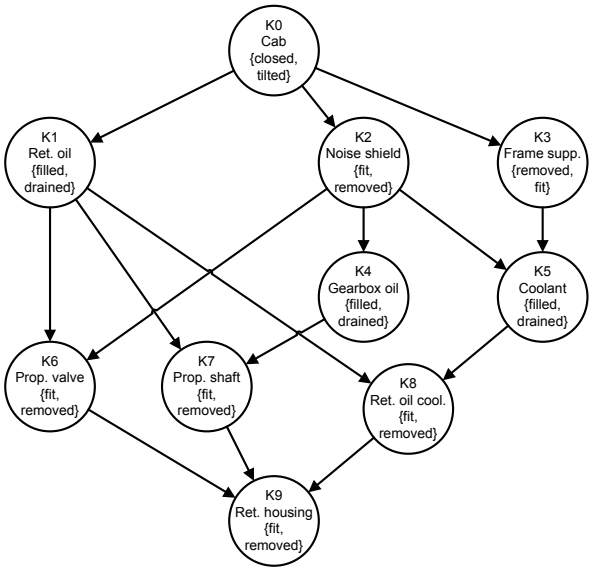


Figure 2: Graph showing dependencies between configuration variables of an auxiliary braking system.

5 Simplifying the Model

The observer's diagnostic engine, D1 in Figure 1, updates the belief state as described in Section 4.2 using the complete model. We argue that better decisions can be made in limited time if the planner uses a simplified model that ignores model features with small probabilities. The model used by the planner's diagnostic engine D2 is obtained by applying a filtering function f_ε with the filtering parameter ε to the parameters p of the current belief state, the CPT:s in the BN, and the probabilities of failed repairs:

$$f_\varepsilon(p) = \begin{cases} 0 & \text{if } p \leq \varepsilon. \\ p & \text{if } \varepsilon < p < 1-\varepsilon. \\ 1 & \text{if } p \geq 1-\varepsilon. \end{cases} \quad (8)$$

Since the probability mass in the belief state must be equal to 1, it is normalized after filtering. In the belief state we only have to enumerate the probabilities of faults and combinations of faults with a probability greater than zero. After filtering, the belief state that the planner is initiated with has a maximum size of $1/\varepsilon$. Also, when a belief state $\mathbf{b}_t$ is updated after an observation $O_i^{t+1}{=}o$ each entry $b_{t+1}(\mathbf{c})$ where $P(O_i{=}o|\mathbf{c}) = 0$ can be removed. Because the time required to update a belief state is dependent of its size, a large ε speeds up planning. However, the parameter ε must be chosen carefully since an ε too large may cause the planner to make suboptimal choices because of lost information.

In the heavy vehicle domain the values of all observation variables from the OBD are observed by the test drive action. The planner can avoid treating all possible outcomes of these observations by clustering them into a single observation variable that indicates when there exists a faulty com-

ponent. The action with the effect observing this variable is the function control.

6 The Planner

The task of the planner is to find a *troubleshooting strategy*, a conditional plan of actions that, if executed to the end, puts the system into a state where all configuration variables are in the mode $assembled$ and the last observation is a function control with a non-indicating outcome. Any troubleshooting strategy π can be seen as a mapping $\pi(s)$ from *system states* to actions. The system state s_t at a given moment t contains sufficient information to describe our knowledge of the system, i.e. the configuration state, the belief state, and the accumulated evidence: $s_t = \langle \mathbf{k}_t, \mathbf{b}_t, \mathbf{e}_{1:t} \rangle$. A troubleshooting strategy π only has actions for non-goal system states that are reachable with π from the initial system state that marks the start point of the conditional plan.

The *expected cost of repair* of a troubleshooting strategy π given an initial system state s, $\mathrm{ECR}(\pi, s)$, is the expected cost of executing π to the end from s:

$$\mathrm{ECR}(\pi, s) = \begin{cases} 0 & \text{if } s \text{ is a goal state.} \\ \mathrm{cost}(\pi(s)) + \\ \sum_{\mathbf{e} \in \mathbf{E}_{\pi(s)}} P(\mathbf{e}|s)\mathrm{ECR}(\pi, s_{\mathbf{e}}) & \text{otherwise.} \end{cases} \tag{9}$$

where $\mathrm{cost}(\pi(s))$ is the cost of performing the action $\pi(s)$, $\mathbf{E}_{\pi(s)}$ is the set of possible outcomes of $\pi(s)$, $P(\mathbf{e}|s)$ is the likelihood of having the outcome $\mathbf{e}$ in s, and $s_{\mathbf{e}}$ is the resulting system state when $\pi(s)$ has the outcome $\mathbf{e}$. The value of $\mathrm{cost}(\pi(s))$ is taken from the action model and $P(\mathbf{e}|s)$ and $s_{\mathbf{e}}$ are retrieved by querying D2.

An optimal troubleshooting strategy π^* is a plan where the expected cost of repair is minimal. For every reachable system state s its expected cost of repair is $\mathrm{ECR}^*(s)$ where

$$\mathrm{ECR}^*(s) = \begin{cases} 0 & \text{if } s \text{ is a goal state.} \\ \min_{a \in A_s} \Big(\mathrm{cost}(a) + \\ \sum_{\mathbf{e} \in \mathbf{E}_a} P(\mathbf{e}|s)\mathrm{ECR}^*(s_{\mathbf{e}}) \Big) & \text{otherwise.} \end{cases} \tag{10}$$

where A_s is the set of actions applicable in s. An action a is applicable in system state s if all its preconditions are fulfilled and if applying it yields a new belief state $\mathbf{b}'$ different from the previous belief state $\mathbf{b}$.

6.1 Composite Actions

Actions are only applicable if they lead to a new belief state, which generally implies gaining new information about the actual state of a vehicle. However, due to the nature of the configuration state space, any configuration state can be reached from any other state – in particular, it must always be possible to reassemble the entire vehicle, after which any other configuration state can be reached by disassembling suitable components. Consequently, allowing the use of arbitrary applicable configuration actions leads to a considerably larger search space.

To avoid this problem, we construct a search space using composite actions. In any given system state $\langle \mathbf{k}, \mathbf{b}, \mathbf{e} \rangle$, an applicable composite action consists of exactly one action a with observe or repair effects that changes the belief state $\mathbf{b}$ to a new belief state $\mathbf{b}' \neq \mathbf{b}$, prefixed by the sequence of configuration actions required to ensure that all preconditions of a are satisfied. Since the configuration state space is reachable, such a sequence can always be found. Thus, composite actions can easily be constructed on demand during search.

6.2 Search Algorithm

The collection of all possible troubleshooting strategies generates an AND/OR graph with alternating layers of OR nodes and AND nodes. Each OR node, including the root, is labeled with a system state and has one AND child for every action applicable in this state. Each AND node, in turn, is labeled with the action that generated the node and has one OR child for every possible outcome of that action. Using the simplified diagnostic engine D2 to expand nodes in a forward search of this graph leads to higher performance and a smaller branching factor.

Suitable algorithms that can find optimal solutions in AND/OR graphs include AO* variants such as LAO* (Hansen and Zilberstein 2001) and algorithms based on dynamic programming such as RTDP (Barto, Bradtke, and Singh 1995) and LDFS (Bonet 2006). All of these algorithms have good anytime properties and can provide suboptimal partial solutions when searching with limited time. Because of the limited time, the decision theoretic approach, where leaf nodes are evaluated by some utility function after an n-step lookahead search, is also suitable for this problem.

If the filtering parameter is chosen such that $\varepsilon \geq \varphi_i$ for all components C_i and repeated observations gain no new information, the search graph is acyclic when only applicable actions are considered. In the experiments, we therefore use the classic version of AO* for acyclic AND/OR graphs as it is described in (Nilsson 1980). LAO* handles cyclic graphs, but has a more complex cost revision phase. An admissible lower bound heuristic function is used to guide the search. In the expansion phase, the unexpanded OR node that can be reached with the highest probability given the current solution is selected for expansion. When timed out, no more nodes are expanded. Instead each encountered unexpanded nodes is considered solved and its cost is set to an upper bound value of the true cost.

6.3 Heuristic Functions

We use an admissible lower bound heuristic function lb where the problem that is described by a system state $s = \langle \mathbf{k}, \mathbf{b}, \mathbf{e} \rangle$ is relaxed by dividing it into $|\mathbf{b}|$ deterministic problems where the true component statuses $\mathbf{c}$ are known. Each of these deterministic problems can easily be solved optimally with a short sequential plan repairing all faulty components and ends with a function control. The cost of this plan $\mathrm{cost}(\pi(\mathbf{c}, \mathbf{k}))$, is weighted by the probability $b(\mathbf{c})$ of the given component status $\mathbf{c}$ to form the lower bound value,

$$lb(s) = \sum_{\mathbf{c} \in \mathbf{b}} b(\mathbf{c})\mathrm{cost}(\pi(\mathbf{c}, \mathbf{k})) \tag{11}$$

The upper bound heuristic function ub that we use when timed out is based on work by (Heckerman, Breese, and Rommelse 1995) and (Langseth and Jensen 2002), but has been significantly extended to provide support for multiple faults as well as configuration states. This heuristic, which requires the existence of a function control action, is an upper bound given that repair actions in the simplified model always succeed. The value of this heuristic is the expected cost of repair using a specific troubleshooting strategy, which inspects components in order of descending ratio between their marginalized probability of being faulty and the cost of inspecting them in the current system state. In other words, components that are likely to be faulty and/or

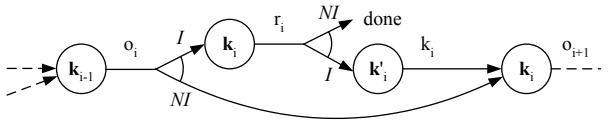


Figure 3: The fixed troubleshooting strategy of the upper bound.

cheap to inspect are treated first. After each repair, a function control is made. If the function control indicates that further components need repair, the system is taken to the same configuration state as after the last inspection and the process is repeated. Each step in this process proceeds as illustrated in Figure 3 beginning in a system state s with the configuration state $\mathbf{k}_{i-1}$, where o_i is a composite action that inspects component i and thereby changes the configuration state from $\mathbf{k}_{i-1}$ to $\mathbf{k}_i$ and r_i is a composite action that repairs component i and performs a function control and thereby changes the configuration state from $\mathbf{k}_i$ to $\mathbf{k}'_i$. To avoid branching, each step must end in a unique configuration state. Therefore, k_i is a composite action that changes the configuration state back from $\mathbf{k}'_i$ to $\mathbf{k}_i$. If a component i cannot be inspected, o_i is instead a composite action that repairs the component and performs a function control and r_i is an empty action.

The costs of the actions are calculated recursively given the configuration states, but the probability that they have to be performed can be taken directly from the belief state of the current system state. Thus, the expected cost of this troubleshooting strategy is

$$ub(s) = \sum_{i=1}^{N} \big(p_{o_i} \cdot \text{cost}(o_i) + p_{r_i} \cdot \text{cost}(r_i) + p_{k_i} \cdot \text{cost}(k_i)\big) \tag{12}$$

where N is the number of components with marginalized probability greater than zero and

$$p_{o_i} = P\Big(\bigvee_{j=i}^{N} C_j = F \Big| \mathbf{b}\Big) \tag{13}$$

$$p_{r_i} = P\big(C_i = F \big| \mathbf{b}\big) \tag{14}$$

$$p_{k_i} = P\Big(C_i = F \wedge \Big(\bigvee_{j=i+1}^{N} C_j = F\Big) \Big| \mathbf{b}\Big) \tag{15}$$

where $\mathbf{b}$ is the belief state of s.

7 Empirical evaluation

We have made a series of empirical experiments to study how the use of a simplified model affects the performance during troubleshooting. Given a specific limit on the amount of time allowed for planning, performance is measured as the average cost of repairing the system when the user follows the recommendations given by the action proposer. The models used in the empirical evaluation are smaller than what a model of an entire heavy vehicle would be. For smaller models we expect that the effect of the use of a simplified model is greater when the time allowed for planning is short. Therefore we have restricted it to 5 seconds only.

7.1 Experimental Setup

In the experiments, three different types of models are used: A true model M^* representing the true system, a full model M_σ used by D1 which is a distorted version of the true model with distortion parameter σ, and a simplified model M_ε used by D2 which is simplified as described in Section 5 using filtering parameter ε. The model M_σ is created by adding noise to the parameters in the Bayesian network of M^* using the log-odds normal distribution as described in (Kipersztok and Wang 2001), i.e. a noise $\omega \sim N(0, \sigma)$ is added to each parameter p in the BN of M^* such that the same parameter p' in the BN of M_σ becomes

$$p' = \frac{1}{1 + (p^{-1} - 1) \cdot 10^{-\omega}} \tag{16}$$

Figure 4 shows the distribution of the distorted parameter p' for different values of σ when $p = 0.2$. When σ grows large the parameters of the BN may receive values far from their original ones. For example when $\sigma = 1.0$ and $p = 0.2$, there is a 10% chance that the new parameter p' becomes greater than 0.8.

For each experiment a belief state $\mathbf{b}^*$ is calculated using M^* given a set of initial evidence. The following steps are then iterated 20 times. First an assignment of component statuses $\mathbf{c}^*$ is drawn randomly from the distribution $\mathbf{b}^*$ which will be considered as the true state of the system. Secondly the model M_σ is generated randomly and M_ε is created from M_σ. Then, until the goal criterion is fulfilled, the action proposer starts recommending actions where the planner is allowed 5 seconds to plan. The outcomes of the recommended actions are drawn randomly in accordance with M^* and $\mathbf{c}^*$ and whenever a repair is made $\mathbf{c}^*$ is updated. After each iteration the costs of all performed actions are summed.

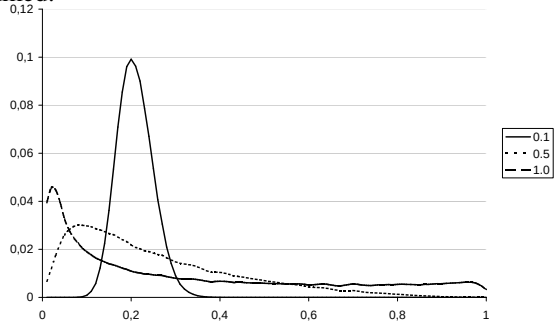


Figure 4: The log-odds normal distribution with mean 0.2 and $\sigma = 0.1$, 0.5 and 1.0

7.2 Models

For the experiments we have used three different models in the heavy vehicles domain with varying difficulty. One of the models is of a real hydraulic braking system of a truck called the retarder. More information on the retarder and how it was modeled can be found in (Pernestål, Warnquist, and Nyberg 2009). This model, M1, has 22 components, 43 observation variables and 10 configuration variables. The cost of actions that repair components varies from 10 to 2200, from filling up with new oil to replacing the electronic control unit. Actions with observe effects cost between 40 and 125 and changes of the configuration state cost between 3 and 332. The observation variables have different specificity and sensitivity. The *specificity* is the probability that the variable is non-indicating when none of its parents in the BN are faulty and the *sensitivity* is the probability that it is indicating when at least one of its parents is faulty. The specificity varies from 0.8–1.0 and the sensitivity from 0.7–0.99.

The other two models, M2 and M3, are synthetic models with similar properties as M1. M2 is a small model with

only 5 components, 7 observation variables, and 4 configuration variables. This model is small enough to allow us to compute optimal and complete conditional plans. M3 is a large model with 50 components, 52 observation variables, and 20 configuration variables. We have created this model to allow us to test the performance of troubleshooting in a larger and more difficult domain. In this model the observation variables belonging to the OBD are made less informative than in M1 and the probability of failed repairs is 0.001. For M1 and M2 this probability is zero.

7.3 Results

In the first experiment we have studied the effect on the average cost of repair when the filtering parameter ε is varied in the range $[0, 0.2]$. In this experiment $\sigma = 0$ and thereby $M_\sigma = M^*$. For the model M2 we have also computed the optimal expected cost of repair when $\varepsilon = 0$ and the time allowed for planning is unlimited. The results are shown in Figure 5.

As expected, filtering has no significant effect on the expected cost of repair M2, which is sufficiently small that the negative effects of ignoring some probabilities tend to be of the same magnitude as the positive effects of being able to explore a larger part of the AND/OR tree. For the larger models M1 and M3, though, we can identify an interval of filtering parameters in which the average cost of repair is considerably reduced compared to the unfiltered case. As ε rises above approximately 0.1, the average cost of repair again begins to rise, as an excessive number of possibilities are removed from the search space.

In the second experiment we have studied the sensitivity to modeling errors in the diagnostic model. We have used $\varepsilon = 0.0001$ and 0.01 and varied the distortion parameter σ in the range [0,1].

Performance does not deteriorate noticeably when the distortion is less than 0.2, but as σ increases above 0.5, the average cost of repair increases significantly, especially for the larger model M3. This is again an expected result, as a distortion factor of 0.5 or greater may result in probabilities very different from the true probabilities in M^*, as seen in Figure 4.

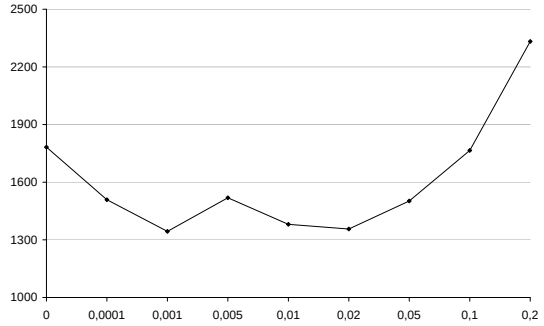
8 Related Work

Since the troubleshooting problem permits planning and acting to be interleaved, general probabilistic planners such as POND (Bryce and Kambhampati 2006) and mGPT (Bonet 2005) that generate complete plans are not suitable when computation time is limited. However, their underlying search algorithms, LAO* and LDFS respectively, can just like AO* be used to create partial plans in anytime. These also have the benefit of being able to find solutions in cyclic search graphs.

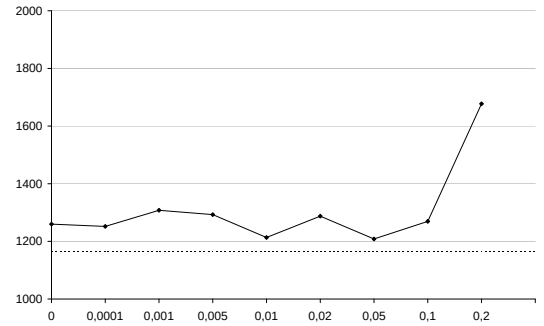
Another strategy of solving complex probabilistic planning problems is to use replanning as in (Yoon, Fern, and Givan 2007). A fast classical planner is used to find a plan that leads to the goal in a deterministic relaxation of the problem. Whenever an unexpected contingency is encountered the plan is revised and in the absence of dead ends the goal is guaranteed to be reached. In this domain, finding any action that takes us closer to the goal is not a problem because the upper bound heuristic already implies a conditional plan that reaches the goal.

In previous work addressing the troubleshooting problem directly, e.g. (de Kleer and Williams 1992), (Sun and Weld 1993), (Heckerman, Breese, and Rommelse 1995), (Langseth and Jensen 2002), the common approach when time is heavily limited is to do a look-ahead search to decide the next action. A look-ahead search is effective in making a decision quickly, but any additional computation time is not used. When instead using an anytime algorithm as in this work the available computation time is used more efficiently.

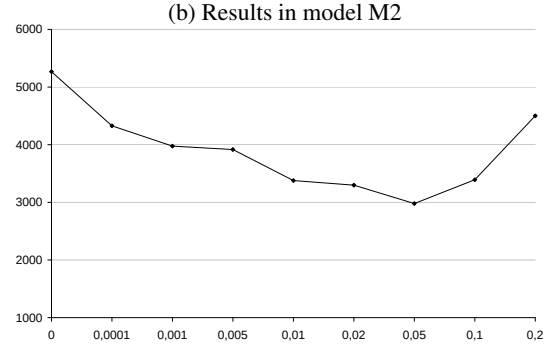
Another approach to speeding up planning for large domains is to reduce the size of the belief state space by restricting it to a set of belief points. This is often done when solving POMDPs, e.g. (Pineau, Gordon, and Thrun 2003). For the troubleshooting problem as it is described here we always have an initial belief state generated from discrete observations. Therefore the reachable belief state space is not continuous and this type of approximation has less effect. The approximations done to the planning model in this work are done to reduce the size of each belief state instead.



(a) Results in model M1



(b) Results in model M2

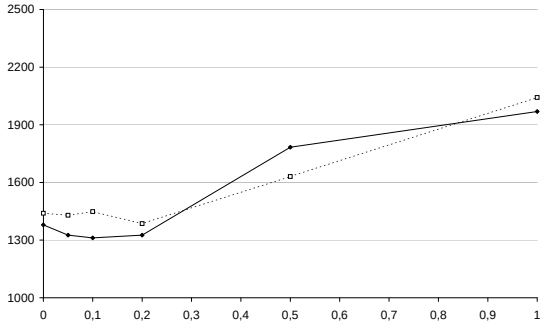


(c) Results in model M3

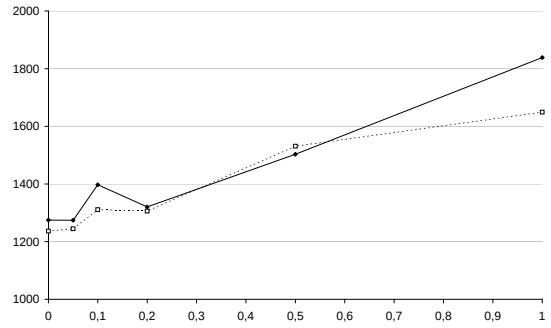
Figure 5: Average cost of repair for different values of ε. The dotted line is the optimal expected cost of repair for M2.

9 Conclusion

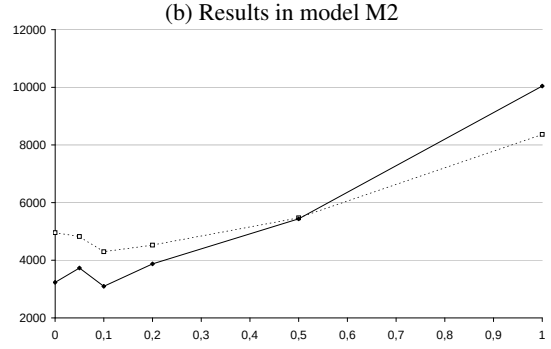
We have studied the problem of troubleshooting mechatronic systems which is a highly relevant problem for the industry and an interesting application of planning. In this problem, planning and acting is interleaved and the time available for computation is limited. We propose that performance can be improved for large problem instances by using a simplified model for belief state prediction during planning. The suggested methodology has been applied to the representative domain of troubleshooting for heavy vehicles. Empirical experiments indicate that by simplifying the model used for planning the performance of troubleshooting is improved when the problem instance is large. This has the potential to lead to significant cost savings when troubleshooting with limited time. When modeling a real mechatronic system one cannot be certain that all parameters in the model are correct. The experiments also show how the performance of the troubleshooting is affected by errors in the parameters of the diagnostic model.



(a) Results in model M1



(b) Results in model M2



(c) Results in model M3

Figure 6: Average cost of repair for different values of σ. The dotted line is when $\varepsilon = 10^{-4}$ and the solid line is when $\varepsilon = 0.01$.

Acknowledgments

This work is supported in part by Scania CV AB, the Vinnova program Vehicle Information and Communication Technology V-ICT, the Center for Industrial Information Technology CENIIT, the Swedish Research Council Linnaeus Center CADICS, and the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII.

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1-2):81–138.

Bonet, B. 2005. mGPT: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research* 24:933–944.

Bonet, B. 2006. Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings, and its application to MDPs. In *Proceedings of ICAPS'06*.

Bryce, D., and Kambhampati, S. 2006. Sequential Monte Carlo in probabilistic planning reachability heuristics. In *Proceedings of ICAPS'06*.

de Kleer, J., and Williams, B. C. 1992. Diagnosis with Behavioral Modes. In *Readings in Model-based Diagnosis*. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Hansen, E. A., and Zilberstein, S. 2001. LAO* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129(1-2):35–62.

Heckerman, D.; Breese, J. S.; and Rommelse, K. 1995. Decision-theoretic troubleshooting. *Communications of the ACM* 38(3):49–57.

Jensen, F. V. 2001. *Bayesian Networks*. New York, NY: Springer-Verlag.

Kipersztok, O., and Wang, H. 2001. Another Look at Sensitivity of Bayesian Networks to Imprecise Probabilities. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*.

Langseth, H., and Jensen, F. V. 2002. Decision theoretic troubleshooting of coherent systems. *Reliability Engineering & System Safety* 80(1):49–62.

Nilsson, N. J. 1980. *Principles of Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann.

Pearl, J. 2000. *Causality*. Cambridge University Press.

Pernestål, A.; Warnquist, H.; and Nyberg, M. 2009. Modeling and Troubleshooting with Interventions Applied to an Auxiliary Truck Braking System. In *Proceedings of 2nd IFAC workshop on Dependable Control of Discrete Systems*.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of IJCAI'03*.

Sun, Y., and Weld, D. S. 1993. A framework for model-based repair. In *Proceedings of AAAI-93*.

U.S Environmental Protection Agency. 2009. On-Board Diagnostics (OBD). www.epa.gov/obd.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *ICAPS*, 352–.